DIGITAL ARITHMETIC
Miloš D. Ercegovac and Tomás Lang
Morgan Kaufmann Publishers, an imprint of Elsevier Science, ©2004
– Updated: December 22, 2003 –

Chapter 7: Solutions to Exercises

– With contributions by Elisardo Antelo –

Exercise 7.1

From

$$\begin{aligned} \epsilon[j] &= 1 - d \cdot R[j] \\ \epsilon[j+1] &= 1 - d \cdot R[j+1] = (\epsilon[j])^2 = (1 - d \cdot R[j])^2 \end{aligned}$$

we get

$$\begin{aligned} 1 - d \cdot R[j+1] &= 1 - 2d \cdot R[j] + (d \cdot R[j])^2 \\ d \cdot R[j+1] &= 2d \cdot R[j] - (d \cdot R[j])^2 \\ R[j+1] &= 2R[j] - d \cdot R[j]^2 = R[j](2 - d \cdot R[j]) \end{aligned}$$

Exercise 7.4

Find the reciprocal of $d = 29/256$ by the multiplicative normalization method. For the maximum error less tha $2^{-12} \approx 0.00024$ in the range $1/2 \leq d < 1$ we scale the input as follows:

$$\frac{1}{d} = \frac{1}{29/256} = \frac{1}{29/32} \times 2^3$$

and compute $\frac{1}{29/32}$

$P[0] = \lfloor 2 - 29/32 \rfloor_4 = 1.0001_2 = 1.0625$

j	$P[j]$	$d[j]$	$R[j]$	$\epsilon[j]$
0	1.0625	0.962891	1.0625	0.037
1	1.037109	0.998623	1.101929	1.38×10^{-3}
2	1.001377	0.999998	1.103446	1.9×10^{-6}
3	1.000002	0.999999	1.103448	3.6×10^{-12}

The answer is $R[3] \times 2^3 = 8.827586...$ compared to $256/29 = 8.827586...$ with an error less than 2^{-12}. Three iterations are used to guarantee that the error is smaller than 2^{-12} for $1/2 \leq d < 1$: for $d = 1/2$, $\epsilon[2] = 3.91 \times 10^{-3} > 2^{-12}$ so another iteration is needed.

Exercise 7.6

Optimal 5-bit input, 4-bit output reciprocal table is shown below. The actual input and output bits are underlined. The case 1.00000 produces the same output as for 1.1111x and needs to be detected.

5-bit input	4-bit output	5-bit input	4-bit output
1.00000	1.00000	1.10000	0.10101
1.00001	0.11111	1.10001	0.10101
1.00010	0.11110	1.10010	0.10100
1.00011	0.11101	1.10011	0.10100
1.00100	0.11100	1.10100	0.10100
1.00101	0.11011	1.10101	0.10011
1.00110	0.11011	1.10110	0.10011
1.00111	0.11010	1.10111	0.10010
1.01000	0.11001	1.11000	0.10010
1.01001	0.11001	1.11001	0.10010
1.01010	0.11000	1.11010	0.10010
1.01011	0.11000	1.11011	0.10001
1.01100	0.10111	1.11100	0.10001
1.01101	0.10111	1.11101	0.10001
1.01110	0.10110	1.11110	0.10000
1.01111	0.10110	1.11111	0.10000

Exercise 7.9

(a) With full multiplier ($55 \times 55 \rightarrow 55$, rounded)

- Rounding error of multiplication: $\pm 2^{-56}$ ($\pm 1/2$ ulp)
- Error due to ones' complement: 2^{-55} (1 ulp)

We now determine the bound on the generated error $\epsilon_G[j]$ by incorporating the bounds of errors associated with each iteration:

$$\begin{aligned} R[j+1] &= R[j](2-(R[j]d \pm 2^{-56}) - 2^{-55}) \pm 2^{-56} \\ &= R[j](2-R[j]d) \mp R[j]2^{-56} - R[j]2^{-55} \pm 2^{-56} \\ &= R[j](2-R[j]d) - \epsilon_G[j] \end{aligned}$$

We assume that $R[j] < 1$ resulting in

$$-2^{-56} + 2^{-55} - 2^{-56} < \epsilon_G[j] < 2^{-56} + 2^{-55} + 2^{-56}$$

That is,

$$0 < \epsilon_G[j] < 2^{-54}$$

To get the final error, we use $\epsilon_T[j] = \epsilon_T[j-1] + \epsilon_G[j]$

$$
\begin{aligned}
-2^{-8} &< \epsilon_T[0] < 2^{8} \\
\epsilon_T[1] &< \epsilon_T[0]^2 + \epsilon_G[0] = 2^{-16} + 2^{-54} \\
\epsilon_T[2] &< (2^{-16} + 2^{-54})^2 + 2^{-54} \\
\epsilon_T[3] &< ((2^{-16} + 2^{-54})^2 + 2^{-54})^2 + 2^{-54} \\
&= (2^{-32} + 2^{-108} + 2^{-69} + 2^{-54})^2 + 2^{-54} = \\
&= 2^{-54} + 2^{-64} + O(2^{-86})
\end{aligned}
$$

(b) With rectangular multiplier ($55 \times 16 \rightarrow 55$, rounded)

j=0

$$
\begin{aligned}
R[1] &= R[0](2 - (R[0]d \mp 2^{-56}) - 2^{-55}) \pm 2^{-16} \\
|\epsilon_G[0]| &\leq 2^{-56} + 2^{-55} + 2^{-16} \\
\epsilon_T[1] &= \epsilon_T[0]^2 + \epsilon_G[0] = (2^{-8})^2 + (2^{-56} + 2^{-55} + 2^{-16}) \\
&= 2^{-15} + 2^{-55} + 2^{-56}
\end{aligned}
$$

j=1

$$
\begin{aligned}
R[2] &= R[1](2 - (R[1]d \mp 2^{-56}) - 2^{-55}) \pm 2^{-32} \\
|\epsilon_G[1]| &\leq 2^{-56} + 2^{-55} + 2^{-32} \\
\epsilon_T[2] &= \epsilon_T[1]^2 + \epsilon_G[1] = (2^{-15} + 2^{-55} + 2^{-56})^2 + 2^{-56} + 2^{-55} + 2^{-32}
\end{aligned}
$$

j=2

$$
\begin{aligned}
R[3] &= R[2](2 - (R[2]d \mp 2 \times 2^{-56}) - 2^{-55}) \pm 2 \times 2^{-32} \\
|\epsilon_G[2]| &\leq 2 \times 2^{-56} + 2^{-55} + 2 \times 2^{-56} = 2^{-54} + 2^{-55} \\
\epsilon_T[3] &= \epsilon_T[2]^2 + \epsilon_G[2] = [(2^{-15} + 2^{-55} + 2^{-56})^2 + 2^{-56} + 2^{-55} + 2^{-32}]^2 \\
&+ 2^{-54} + 2^{-55} \\
&= 2^{-54} + 2^{-55} + 2^{-60} + O(2^{-64})
\end{aligned}
$$

Exercise 7.13

$x = 1310/4096 = 0.010100011110$, $d = 2883/4096 = 0.101101000011$

The initial value: $R[0] = 2.98 - d = 1.100100101000$. As indicated on p.373, the maximum relative error is about 10^{-1}. For an error of 2^{-12}, two iterations are sufficient.

a) Using Newton-Raphson method (results truncated to 12 fractional bits):

j	$R[j]$	$\epsilon[j]$
0	1.100100101000	-0.107
1	1.011001111001	0.011
2	1.011010111010	1.3×10^{-4}

The error in the computed quotient $q = x \times R[2] = 0.011101000100$ is smaller than 6×10^{-5} which is less than 2^{-12}.

b) Using multiplicative method: $P[0] = 2.98 - 2d = 1.5722 = 1.100100101000$ (Results truncated to 12 bits)

- Step 1:
  $d[0] = d \cdot P[0] = 1.000110110100$; $q[0] = x \cdot P[0] = 0.100000001011$
- Step 2:
  $P[1] = 2 - d[0] = 0.111001001011$
  $d[1] = d[0] \cdot P[1] = 0.1111111010001$; $q[1] = q[0] \cdot P[1] = 0.011100110000$
- Step 3:
  $P[2] = 2 - d[1] = 1.000000101110$;
  $d[2] = d[1] \cdot P[2] = 0.111111111111$; $q[2] = q[1] \cdot P[2] = 0.011101000100$
  Again, the error in the computed quotient is less than 2^{-12}.

The error in the quotient is 5.9×10^{-5}.

Exercise 7.17

The algorithm to implement is:

$$X[0] = x, \quad S[0] = x, \quad P[0] = A$$

where A is an approximation to $1/\sqrt{x}$ with an error less than 2^{-8}.

for $j = 0$ to 3
$P[j] = 1 + \frac{1}{2}(1 - X[j])$
$P2[j] = P[j]P[j]$
$X[j+1] = X[j]P2[j]$
$S[j+1] = S[j]P[j]$

(a) Alternative with a full 55×55 multiplier, a 3-stage pipeline.

- $P[0] = A$ – one cycle;
- Scheduling of an iteration in the pipelined multiplier is shown in Figure E7.17. It takes 4 cycles to obtain $S[j+1]$. An iteration takes 6 cycles.
- Latency:

  1 cycle for initial approximation
  3 full iterations, each 6 cycles for a total of 18 cycles
  partial iteration to obtain $S[4]$ in 4 cycles
  total: 23 cycles

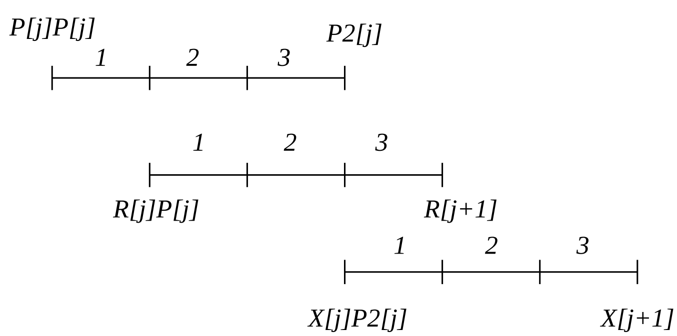

Figure E7.17: Scheduling of one iteration

(b) With 55×16 rectangular multipliers (single stage)

- $P[0] = A$, a 9-bit approximation; 1 cycle
- First iteration:

  $x[1] = x[0] \cdot P[0]$; (55×9); 1 cycle
  $x[1] = x[1] \cdot P[0]$; (55×9); 1 cycle
  $S[1] = S[0] \cdot P[0]$; (55×9); 1 cycle
- Second iteration:

$P[1] = 1 + \frac{1}{2}(1 - x[1])$; rounded to 16 bits
$x[2] = x[1] \cdot P[1]$; (55×16); 1 cycle
$x[2] = x[2] \cdot P[1]$; (55×16); 1 cycle
$S[2] = S[1] \cdot P[1]$; (55×16); 1 cycle

– Third iteration:

$P[2] = 1 + \frac{1}{2}(1 - x[2])$; rounded to 32 bits
$x[3] = x[2] \cdot P[2]$; (55×32); 2 cycles
$x[3] = x[3] \cdot P[2]$; (55×32); 2 cycles
$S[3] = S[2] \cdot P[2]$; (55×32); 2 cycles

– Termination:

$P[3] = 1 + \frac{1}{2}(1 - x[3])$; rounded to 55 bits
$S[4] = S[3] \cdot P[3]$; (55×55); 4 cycles

– Latency: 1+3+3+6+4 = 17 cycles. This can be reduced to 13 cycles if two rectangular multipliers are used.